Programming Exercise 06
Military Unit (Implementing Inheritance)

C# Step by Step

This activity consists of four programming exercises. The following exercises are open book and open note. You are free to use any written documentation you wish. However, these are individual exercises, and you cannot consult with each other in writing your programs.

This programming exercise has four parts consisting of four requirements. The grade for each requirement is indicated, for a maximum of 100 points. You have two options for the first part. Select one. At a minimum, your program must compile successfully and run.

Using inheritance, abstraction, classes and objects, implement a military unit. Create super or abstract classes representing several components of a military unit. Examples might include personnel, weapons, equipment, vehicles, missions, etc. Each parent class should be subclassed several times. Thinking about test first development, you may want to write the main program first, before you create any other classes or objects. That way, you can create classes with the appropriate properties and methods, and instantiate objects as needed to carry out the mission.

First part: 70 points Choose one option.

Option 1, main class first Write your main class first. This will consist of your variables and methods that execute the mission of your unit. Since you have not built your unit, you cannot actually run the program, so you will have to supply your methods with appropriate print statements to have your programn run to completion.

Option 2, parent classes first Write your parent classes first. These will be the classes you will subclass to create the appropriate objects. For example, you might have a personnel class, a weapons class, a missions class, and a vehicles class.

Second part: 80 points If you chose to implement **Option 1** in the first part, implement **Option 2** in in this part. If you chose to implement **Option 2** in the first part, in this part, implement the appropriate subclasses. For example, if one of your parent classes is `Weapon`, in this class you may elect to implement `SmallCaliberWeapon`, `IndirectFireWeapon`, and `DirectFireWeapon`.

Third part: 90 points If you chose to implement **Option 2** in the first part, in this part implement the appropriate methods as virtual and overidden methods, or use another suitable technique. As an example, if you have a method named `mount()`, it will exhibit different behavior depending on whether it's called on a type `RifleSight` or a type `Helicopter`. As another example, if you have a method named `load()`, it will exhibit different behavior depending on whether it's called on a type `AutomaticWeapon` or a type `Howitzer`. Otherwise, implement your subclasses.

Fourth part: 100 points If you chose to implement **Option 2** in the first part, in this part implement **Option 1** in this part. Otherwise, implement the methods as instructed for the third part.

Note that you may chose to implement the main class first and the data type classes second, or the data type classes first and the main class second. In both cases, you need a main class to instantiate objects of the data types you create and exercise the methods of those data types. Generally most developers will implement the data types first and the main class second. Doing it the other way is part of a discipline called *test driven development*, and is associated with agile techniques. Regardless of what you choose, you should be familiar with both processes.